A Novel Training Algorithm based on Limited-Memory quasi-Newton Method with Nesterov's Accelerated Gradient in Neural Networks and its Application to Highly-Nonlinear Modeling of Microwave Circuit

Shahrzad MAHBOUBI† and Hiroshi NINOMIYA††

Graduate school of Electrical and Information Engineering, Shonan Institute of Technology
Email: 18T2012@sit.shonan-it.ac.jp†, ninomiya@info.shonan-it.ac.jp††

***Abstract*—This paper describes a novel algorithm based on Limited-memory quasi-Newton method incorporating Nesterov's accelerated gradient for faster training of neural networks. Limited-memory quasi-Newton is one of the most efficient and practical algorithms for solving large-scale optimization problems. Limited-memory quasi-Newton is also the gradient-based algorithm using the limited curvature information without the approximated Hessian such as the normal quasi-Newton. Therefore, Limited-memory quasi-Newton attracts attention as the training algorithm for large-scale and complicated neural networks. On the other hand, Nesterov's accelerated gradient method has been widely utilized as the first-order training algorithm for neural networks. This method accelerated the steepest gradient method using the inertia term for the gradient vector. In this paper, it is confirmed that the inertia term is effective for the acceleration of Limited-memory quasi-Newton based training of neural networks. The acceleration of the proposed algorithm is demonstrated through the computer simulations compared with the conventional training algorithms for a benchmark problem and a real-world problem of the microwave circuit modeling.**

Keywords–Neural networks; training algorithm; Limited-memory quasi-Newton method; Nesterov's accelerated gradient method; highly-nonlinear function modeling.

I. INTRODUCTION

This paper extends our previous work [1], presented at the IARIA FUTURE COMPUTING 2018, on acceleration of Limited-memory quasi-Newton based training of neural networks.

Neural networks have been recognized as a useful tool for function approximation problems. Especially, neural networks can efficiently approximate functions with highly-nonlinear input-output properties [2]. For example, neural networks can be utilized as the microwave circuit modeling in which the network is trained from Electro-Magnetic (EM) data over a range of geometrical parameters and trained networks become models providing fast solutions of the EM behavior it learned [3]-[6]. Generally, EM behaviors for geometrical behaviors are highly-nonlinear [3]. This is useful for modeling where formulas are not available or original models are computationally too expensive.

Training is the most important step in developing a neural network model. Gradient-based algorithms are popularly used for the training and can be divided into two categories: first-order methods and second or approximated second order methods [2]. The formers are popularly used for this purpose [8]-[14]. The typical first-order training is the steepest gradient descant method so-called Backpropagation (BP) [2]. BP was accelerated by the momentum (inertia) term [8]. This technique was referred to as Classical Momentum method (CM). A simple modification to improve the performance of CM was introduced as Nesterov's Accelerated Gradient method (NAG) [7][8]. On the other hand, the training algorithms need strategies to determine *stepsize* or *learning rate* and the efficiency of training is highly dependent on the stepsize. Adaptive gradient method (AdaGrad) [9] and Resilient Mean Square backpropagation (RMSprop) [12] were introduced for the neural network training with the adaptive stepsize. Moreover, the combination algorithm of the momentum acceleration and the adaptive stepsize was Adam [14]. The recent developments of training were mostly based on the stochastic strategies such as the minibatch methods in which the gradients were calculated using the portion of all training samples. However, stochastic strategies are not suitable for the neural network training with highly-nonlinear properties [15]. Therefore, the full batch strategies are focused on this paper. Note that to the best of author's knowledge, the convergence for the non-convex problems such as the neural network training was only discussed for Adam of full batch strategies [16].

Adam is the most popular and effective first-order algorithm. With the progress of AI and IoT technologies, however, the characteristics between inputs and desired outputs of the training samples have become more complex. For such scenarios, neural networks need to deal with highly-nonlinear functions. Under such circumstances, the first order methods converge too slowly and optimization error cannot be effectively reduced within finite time in spite of its advantage [17]. The second and approximated second order methods are represented by Newton and quasi-Newton (QN) methods, respectively. Particularly, the QN training, which is one of the most effective optimization [18] is widely utilized as the robust training algorithm for highly-nonlinear function approximations [3]-[6]. However, the QN iteration includes the product matrix (the approximated Hessian) and vector, that is QN needs the massive computer resources of memories as the scale of neural network becomes larger. QN incorporating Limited-memory scheme so-called Limited-memory QN (LQN) is effective for solving large-scale problems whose Hessian matrices cannot be computed at a reasonable cost or is not sparse [18][19]. Furthermore, the momentum acceleration of QN was introduced as Nesterov's accelerated quasi-Newton method (NAQ) [17]. It was shown that the inertia term was effective to reduce the number of iterations of QN and to accelerate its convergence speed.

In this paper, the acceleration technique of LQN is proposed using Nesterov's accelerated gradient [1]. First of all,

a novel algorithm is derived from the detailed consideration of the derivation process of NAQ. The proposed algorithm, which is referred to as Limited-memory NAQ (LNAQ), is accelerated incorporating the momentum acceleration scheme of NAQ into LQN. The proposed training is demonstrated through the computer simulations. The effectiveness of the inertia term is confirmed by the comparison of LNAQ with LQN using a benchmark problem of highly-nonlinear function approximation. Finally, it is shown that the proposed algorithm is efficient and practical for the real-world problem of microwave circuit modeling.

The contents of this paper is structured as follows: Section II shows the related works. Section III Introduces the formulation of training and conventional gradient-based algorithms such as BP, CM, NAG, AdaGrad, RMSprop, Adam and LQN. Section VI proposes the novel algorithm - LNAQ, which is the acceleration method of LQN by introducing momentum coefficient. Section V provides simulation results in order to demonstrate the validity of the proposed LNAQ. Section VI concludes this paper and describes the future works.

II. Related work

Recently, the neural networks having deep and huge strictures have attracted enormous research attentions in pattern recognition, computer vision, and speech recognition [20]. First-order techniques such as CM, NAG, Adagrad, RMSprop, and Adam [7]-[9][12][14], have been used mostly for training of deep neural networks. On the other hand, neural network techniques have been recognized as useful tool for modeling and design optimaization problems in analog and microwave circuits design of CAD [3]-[6][21]-[43] in which their I/O characteristics are strongly nonlinear. For example, EM behavior modeling [3]-[6], analog integrated circuits (IC) [23]-[26], oscillation [27][28], antenna applications [29], nonlinear microwave circuit optimization [30]-[32], waveguide filters [33]-[36], low-pass filters [17][36]-[38][49], power amplifier modeling [39]-[42], vias and interconnects [43], have been studied. Neural networks can be used for developing new models whose formula are not available or original models are computationally too expensive. In these studies, the neural networks with deep structure are not necessarily utilized. Suitable training algorithms for these purposes are approximated second-order methods such as QN and Levenberg-Marquardt method (LM). These methods produce models with lower training error and have faster speed of training than first-order methods [5]. LM method is a modified version of the Gauss-Newton method (GN). Particularly, LM can be thought of as a combination of the strong convergence ability of Steepest Gradient method and the rapid convergence speed of GN [44]-[46]. However, LM needs to solve the system of linear equations in each iteration [19]. On the contrary QN iterates approximating the inverse matrix of Hessian [18][19]. Therefore, QN did not need the matrix solution in each iteration, but had to handle the variable-metric matrix. As a result, these algorithms were unsuitable for training large-scale neural networks with much small errors. That is, for modeling of large-scale problems with highly-nonlinearity, the matrix handing has not reasonable cost [18]. Therefore, LQN was used for the training [18][19]. The main idea of LQN is to use curvature information from only the most recent iterations to construct the Hessian approximation. Curvature information from earlier iteration, which is less likely to be relevant to the actual behavior of the Hessian at the current iteration, is discarded in the interest of saving memory [18].

On the other hand, the acceleration algorithm of QN was proposed as NAQ in [17]. This method can have realized introducing momentum coefficient into QN and drastically improved the convergence speed of the QN. As far as the author's best knowledge, NAQ was the first acceleration technique of QN using the momentum term.

III. Formulation of Training and Gradient-based Training Methods

A. Formulation of training

Let $\mathbf{d}_p$, $\mathbf{o}_p$, and $\mathbf{w} \in \mathbb{R}^D$ be the p-th desired, output, and weight vectors, respectively. The error function $E(\mathbf{w})$ is defined as the mean squared error (MSE) of

$$E(\mathbf{w}) = \frac{1}{|T_r|} \sum_{p \in T_r} E_p(\mathbf{w}), \quad E_p(\mathbf{w}) = \frac{1}{2} \|\mathbf{d}_p - \mathbf{o}_p\|^2, \quad (1)$$

where T_r denotes a training data set $\{\mathbf{x}_p, \mathbf{d}_p\}, p \in T_r$ and $|T_r|$ is the number of training samples. Among the gradient-based algorithms, (1) is minimized by

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}, \quad (2)$$

where k is the iteration count and $\mathbf{v}_{k+1}$ is the update vector. A simple gradient descent algorithm that is the original BP [2] has

$$\mathbf{v}_{k+1} = -\alpha_k \nabla E(\mathbf{w}_k), \quad (3)$$

where α_k is the stepsize and $\nabla E(\mathbf{w}_k)$ is the gradient vector at $\mathbf{w}_k$.

B. First-order training methods

This section introduces the conventional first-order training methods.

1) Classical Momentum method (CM)

CM is a technique for accelerating BP that accumulates a previous update vector in directions of persistent reduction in the training [8]. The update vector of CM is given by

$$\mathbf{v}_{k+1} = \mu \mathbf{v}_k - \alpha_k \nabla E(\mathbf{w}_k), \quad (4)$$

where $0 \leq \mu \leq 1$ denotes the momentum coefficient.

2) Nesterov's Accelerated Gradient method (NAG)

NAG has been the subject of much recent studies in machine learning [7][8][17]. Arguing that NAG can be viewed as a simple modification of CM, and can sometimes provide a distinct improvement in performance for acceleration of neural network training [8]. CM computes the gradient vector from the current position $\mathbf{w}_k$, whereas NAG first performs a partial update to $\mathbf{w}_k$, computing $\mathbf{w}_k + \mu \mathbf{v}_k$, and then computes the

gradient at $\mathbf{w}_k + \mu\mathbf{v}_k$. NAG have better convergence rate than CM [7]. NAG update can be written as

$$\mathbf{v}_{k+1} = \mu_k\mathbf{v}_k - \alpha_k\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k), \tag{5}$$

where $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$ means the gradient of $E(\mathbf{w})$ at $\mathbf{w}_k + \mu\mathbf{v}_k$ and is referred to as Nesterov's accelerated gradient vector.

3) Adaptive Gradient method (AdaGrad)

AdaGrad [9] is the first-order gradient-based training algorithms with an adaptive stepsize. The update vector of AdaGrad is given by

$$v_{k+1,i} = -\frac{\alpha}{\sqrt{\sum_{s=1}^{k}(\nabla E(\mathbf{w}_s)_i)^2}}\nabla E(\mathbf{w}_k)_i. \tag{6}$$

Here $v_{k+1,i}$ and $\nabla E(\mathbf{w}_k)_i$ are the *i*-th elements of $\mathbf{v}_{k+1}$ and $\nabla E(\mathbf{w}_k)$, respectively. α is a global stepsize shared by all dimensions. The recommended value of α is $\alpha = 0.01$ [9][10][11].

4) Resilient Mean Square backpropagation method (RMSprop)

RMSprop [12] was a mini-batch version of Rprop [10]. The update vector of RMSprop is

$$v_{k+1,i} = -\frac{\alpha}{\sqrt{\theta_{k,i} + \lambda}}\nabla E(\mathbf{w}_k)_i, \tag{7}$$

where $\lambda = 10^{-8}$ and

$$\theta_{k,i} = \gamma\theta_{k-1,i} + (1-\gamma)(\nabla E(\mathbf{w}_k)_i)^2. \tag{8}$$

$\theta_{k,i}$ is the parameter of *k*-th iteration and i-th element. γ and the global stepsize of α are set to 0.9 and 0.001, respectively in [12].

5) Adam

Adam is the most popular and effective gradient-based training algorithm with less memory requirement [14]. Adam was realized by combining RMSprop with CM. The update vector of Adam can be written as

$$v_{k+1,i} = -\alpha\frac{\hat{m}_{k,i}}{(\sqrt{\hat{\theta}_{i,k}} + \lambda)}, \tag{9}$$

where

$$\hat{m}_{k,i} = \frac{m_{i,k}}{(1-\gamma_1^k)}, \tag{10}$$

and

$$\hat{\theta}_{k,i} = \frac{\theta_{k,i}}{(1-\gamma_2^k)}. \tag{11}$$

Here, $m_{k,i}$ and $\theta_{k,i}$ are given by

$$m_{k,i} = \gamma_1 m_{k-1,i} + (1-\gamma_1)\nabla E(\mathbf{w}_k)_i, \tag{12}$$

and

$$\theta_{k,i} = \gamma_2\theta_{k-1,i} + (1-\gamma_2)(\nabla E(\mathbf{w}_k)_i)^2, \tag{13}$$

where $\lambda = 10^{-8}$ and γ_1^k and γ_2^k denote the *k*-th power of γ_1 and γ_2, respectively. α is the global stepsize and the recommended value is $\alpha = 0.001$ [14]. $m_{k,i}$ and $\theta_{k,i}$ are *i*-th elements of the gradient and the squared gradient, respectively. The hyper-parameters $0 \leq \gamma_1, \gamma_2 < 1$ control the exponential decay rates of these running averages. The running average themselves are estimates of the first (the mean) moment and the second raw (the uncentered variance) moment of the gradient. γ_1 and γ_2 are chosen to be 0.9 and 0.999, respectively in [14]. All operations on vectors are element-wise.

Note that the recent developments of the training algorithm such as AdaGrad, RMSprop and Adam were based on the stochastic strategies. These strategies are not suitable for the training of highly-nonlinear function modeling [5][15]. Therefore, we focus on the methods using the curvature information and the full batch strategy in this paper.

C. Limited-memory quasi-Newton method (LQN)

QN method is the efficient optimization algorithm using the curvature information and commonly used as training method for highly-nonlinear function problems. The update vector of QN is defined as

$$\mathbf{v}_{k+1} = \alpha_k\mathbf{c}_k, \tag{14}$$

where

$$\mathbf{c}_k = -\mathbf{H}_k\nabla E(\mathbf{w}_k), \tag{15}$$

$\mathbf{c}_k$ is the direction vector and $\mathbf{H}_k$ is a symmetric positive definite matrix. $\mathbf{H}_k$ is iteratively given by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula of (16) as the approximated inverse matrix of Hessian [18][19].

$$\begin{aligned}\mathbf{H}_{k+1} = \mathbf{H}_k &- \frac{(\mathbf{H}_k\mathbf{y}_k)\mathbf{s}_k^{\mathrm{T}} + \mathbf{s}_k(\mathbf{H}_k\mathbf{y}_k)^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k} \\ &+ \left(1 + \frac{\mathbf{y}_k^{\mathrm{T}}\mathbf{H}_k\mathbf{y}_k}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}\right)\frac{\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k},\end{aligned} \tag{16}$$

where

$$\mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k, \tag{17}$$

$$\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k) + \xi_k\mathbf{s}_k = \epsilon_k + \xi_k\mathbf{s}_k, \tag{18}$$

and ξ_k is defined as

$$\xi_k = \omega \parallel \nabla E(\mathbf{w}_k) \parallel + max\{-\epsilon_k^{\mathrm{T}}\mathbf{s}_k/\parallel \mathbf{s}_k \parallel^2, 0\}, \tag{19}$$

$$\begin{cases}\omega = 2 & if \parallel \nabla E(\mathbf{w}_k) \parallel^2 > 10^{-2}, \\ \omega = 100 & if \parallel \nabla E(\mathbf{w}_k) \parallel^2 < 10^{-2}.\end{cases} \tag{20}$$

Here, ξ_k was introduced to guarantee the numerical stability and the global convergence [47]. For the purpose of reducing the amount of computer resources used in QN, a sophisticated technique incorporating the limited-memory scheme is widely utilized for the calculation of $\mathbf{v}_{k+1}$ as LQN. Specifically, this method is useful for solving problems whose $\mathbf{H}_k$ in (16) cannot be computed at a reasonable cost. That is, instead of storing $D \times D$ matrix of $\mathbf{H}_k$, only $2 \times t \times D$ elements have to be stored. Furthermore, the product of the matrix and vector can be changed to only the inner product of stored vectors. Here, D is the dimension of $\mathbf{w}$ and $t(\ll D)$ is a hyper-parameter defined by user. That is, $\mathbf{s}_i$ and $\mathbf{y}_i$ vectors between $i = k$ and $i = k-t$ are stored in LQN. As a result, the computational resources of memory and calculation costs are drastically reduced when $t \ll D$ [18]. The LQN scheme is illustrated in Algorithms 1 and 2. In Algorithm 1, α_k is derived using the line search in which Armijo's condition of (21) is utilized.

$$E(\mathbf{w}_k + \alpha_k \mathbf{c}_k) \leq E(\mathbf{w}_k) + \chi \alpha_k \nabla E(\mathbf{w}_k)^{\mathrm{T}} \mathbf{c}_k, \tag{21}$$

where $0 < \chi < 1$ and $\chi = 0.001$ in this paper.

Algorithm 1: Limited-memory quasi-Newton (LQN)

1. $k = 1$;
2. $\mathbf{w}_1 = rand[-0.5, 0.5]$(uniform random numbers);
3. Calculate $\nabla E(\mathbf{w}_1)$;
4. **While**$(k < k_{max})$
   (a) Calculate the direction vector $\mathbf{c}_k$ using Algorithm 2;
   (b) Calculate stepsize α_k using Armijo's condition;
   (c) Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{c}_k$;
   (d) Calculate $\nabla E(\mathbf{w}_{k+1})$;
   (e) $k = k + 1$;
5. **return** $\mathbf{w}_k$;

Algorithm 2: Direction Vector of LQN

1. $\mathbf{c}_k = -\nabla E(\mathbf{w}_k)$;
2. for $i : k, k-1, \ldots, k - \min(k, (t-1))$;
   (a) $\beta_i = \mathbf{s}_i^{\mathrm{T}} \mathbf{c}_k / \mathbf{s}_i^{\mathrm{T}} \mathbf{y}_i$;
   (b) $\mathbf{c}_k = \mathbf{c}_k - \beta_i \mathbf{y}_i$;
3. if $k > 1, \mathbf{c}_k = (\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k / \mathbf{y}_k^{\mathrm{T}} \mathbf{y}_k) \mathbf{c}_k$;
4. for $i : k - \min(k, (t-1)), \ldots, k-1, k$;
   (a) $\tau = \mathbf{y}_i^{\mathrm{T}} \mathbf{c}_k / \mathbf{y}_i^{\mathrm{T}} \mathbf{s}_i$;
   (b) $\mathbf{c}_k = \mathbf{c}_k - (\beta_i - \tau) \mathbf{s}_i$;
5. **return** $\mathbf{c}_k$;

IV. Proposed Algorithm - Limited-memory Nesterov's Accelerated quasi-Newton method (LNAQ)

NAQ training was derived from the quadratic approximation of (1) around $\mathbf{w}_k + \mu \mathbf{v}_k$ whereas QN used the approximation of (1) around $\mathbf{w}_k$ [17]. NAQ drastically improved the convergence speed of QN using the gradient vector at $\mathbf{w}_k + \mu \mathbf{v}_k$ of $\nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$ called Nesterov's accelerated gradient vector [7][17]. This means that the inertia term of $\mu \mathbf{v}_k$ was effective to accelerate the QN. First of all, the derivation of NAQ is briefly introduced as follows:

Let $\Delta \mathbf{w}$ be the vector $\Delta \mathbf{w} = \mathbf{w} - (\mathbf{w}_k + \mu_k \mathbf{v}_k)$, the quadratic approximation of (1) around $\mathbf{w}_k + \mu_k \mathbf{v}_k$ is defined as

$$\begin{aligned} E(\mathbf{w}) \simeq E(\mathbf{w}_k + \mu_k \mathbf{v}_k) + \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)^{\mathrm{T}} \Delta \mathbf{w} \\ + \frac{1}{2} \Delta \mathbf{w}^{\mathrm{T}} \nabla^2 E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \Delta \mathbf{w}, \end{aligned} \tag{22}$$

where $\nabla^2 E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ is Hessian of $E(\mathbf{w})$. The minimizer of this quadratic function is explicitly given by

$$\Delta \mathbf{w} = -\nabla^2 E(\mathbf{w}_k + \mu_k \mathbf{v}_k)^{-1} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k). \tag{23}$$

Then the new iterate is defined as

$$\begin{aligned} \mathbf{w}_{k+1} = (\mathbf{w}_k + \mu_k \mathbf{v}_k) \\ -\nabla^2 E(\mathbf{w}_k + \mu_k \mathbf{v}_k)^{-1} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k). \end{aligned} \tag{24}$$

This iteration is considered as Newton method with the momentum term $\mu \mathbf{v}_k$. Here Hessian $\nabla^2 E(\mathbf{w}_k + \mu \mathbf{v}_k)$ is approximated by $\hat{\mathbf{B}}_{k+1}$ and the rank-2 updating formula of this matrix is derived. Let $\mathbf{p}_k$ and $\mathbf{q}_k$ be

$$\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k), \tag{25}$$

$$\mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k), \tag{26}$$

and the secant condition is defined as

$$\mathbf{q}_k = \hat{\mathbf{B}}_{k+1} \mathbf{p}_k. \tag{27}$$

The suitable rank-2 updating formula for $\hat{\mathbf{B}}_{k+1}$ is derived as follows. The matrix $\hat{\mathbf{B}}_{k+1}$ is defined using arbitrary vectors $\mathbf{t}$ and $\mathbf{u}$ and constants a and b as

$$\hat{\mathbf{B}}_{k+1} = \hat{\mathbf{B}}_k + a \mathbf{t}\mathbf{t}^{\mathrm{T}} + b \mathbf{u}\mathbf{u}^{\mathrm{T}}. \tag{28}$$

Substitute (28) into the secant condition (27),

$$\begin{aligned} \mathbf{q}_k = \left(\hat{\mathbf{B}}_k + a \mathbf{t}\mathbf{t}^{\mathrm{T}} + b \mathbf{u}\mathbf{u}^{\mathrm{T}} \right) \mathbf{p}_k = \\ \hat{\mathbf{B}}_k \mathbf{p}_k + a \mathbf{t} \left(\mathbf{t}^{\mathrm{T}} \mathbf{p}_k \right) + b \mathbf{u} \left(\mathbf{u}^{\mathrm{T}} \mathbf{p}_k \right). \end{aligned} \tag{29}$$

Since $\mathbf{t}^{\mathrm{T}} \mathbf{p}_k$ and $\mathbf{u}^{\mathrm{T}} \mathbf{p}_k$ are scalars, both of conditions $\mathbf{t} = \mathbf{q}_k$ and $\mathbf{u} = -\hat{\mathbf{B}}_k \mathbf{p}_k$ are necessary to the secant condition of (27). Furthermore scalars a and b are given by $a \left(\mathbf{t}^{\mathrm{T}} \mathbf{p}_k \right) = 1$ and $b \left(\mathbf{u}^{\mathrm{T}} \mathbf{p}_k \right) = 1$, respectively. As a result, the rank-2 updating formula for NAQ is defined as

$$\hat{\mathbf{B}}_{k+1} = \hat{\mathbf{B}}_k + \frac{\mathbf{q}_k \mathbf{q}_k^{\mathrm{T}}}{\mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k} - \frac{\hat{\mathbf{B}}_k \mathbf{p}_k \mathbf{p}_k^{\mathrm{T}} \hat{\mathbf{B}}_k}{\mathbf{p}_k^{\mathrm{T}} \hat{\mathbf{B}}_k \mathbf{p}_k}. \tag{30}$$

Next, it is shown that $\hat{\mathbf{B}}_{k+1}$ of (30) is the symmetric positive definite matrix under the *Definition*: $\hat{\mathbf{B}}_k$ *is the symmetric positive definite matrix*. Here the following conditions are guaranteed for the above:

(a): $\hat{\mathbf{B}}_{k+1}$ of (30) satisfies the secant condition $\mathbf{q}_k = \hat{\mathbf{B}}_{k+1}\mathbf{p}_k$.

(b): If $\hat{\mathbf{B}}_k$ is symmetry, $\hat{\mathbf{B}}_{k+1}$ is also symmetry.

(c): If $\hat{\mathbf{B}}_k$ is the positive definite matrix, $\hat{\mathbf{B}}_{k+1}$ is also the positive definite matrix.

Proof of (a):

From (30) the secant condition $\mathbf{q}_k = \hat{\mathbf{B}}_{k+1}\mathbf{p}_k$:

$$\begin{aligned}\hat{\mathbf{B}}_{k+1}\mathbf{p}_k &= \left(\hat{\mathbf{B}}_k + \frac{\mathbf{q}_k\mathbf{q}_k^{\mathrm{T}}}{\mathbf{q}_k^{\mathrm{T}}\mathbf{p}_k} - \frac{\hat{\mathbf{B}}_k\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{B}}_k}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{B}}_k\mathbf{p}_k}\right)\mathbf{p}_k \\ &= \hat{\mathbf{B}}_k\mathbf{p}_k + \frac{\mathbf{q}_k\mathbf{q}_k^{\mathrm{T}}}{\mathbf{q}_k^{\mathrm{T}}\mathbf{p}_k}\mathbf{p}_k - \frac{\hat{\mathbf{B}}_k\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{B}}_k}{\mathbf{p}_k^{\mathrm{T}}\hat{\mathbf{B}}_k\mathbf{p}_k}\mathbf{p}_k = \mathbf{q}_k\end{aligned} \tag{31}$$

□

Proof of (b): This is clear from (30).

□

Proof of (c):

First, $\mathbf{q}_k^{\mathrm{T}}\mathbf{p}_k > 0$ will be shown. When the stepsize α_k is calculated by the exact line search, that is,

$$dE\left(\mathbf{w}_{k+1}\right)/d\alpha_k = -\nabla E\left(\mathbf{w}_{k+1}\right)^{\mathrm{T}}\hat{\mathbf{H}}_k\nabla E\left(\mathbf{w}_k + \mu\mathbf{v}_k\right) = 0. \tag{32}$$

As a result,

$$\mathbf{q}_k^{\mathrm{T}}\mathbf{p}_k = \alpha_k\nabla E\left(\mathbf{w}_k + \mu\mathbf{v}_k\right)^{\mathrm{T}}\hat{\mathbf{H}}_k\nabla E\left(\mathbf{w}_k + \mu\mathbf{v}_k\right) > 0, \tag{33}$$

is derived. It is guaranteed in (33) that $\hat{\mathbf{H}}_k$ is the positive definite matrix because it is the inverse matrix of $\hat{\mathbf{B}}_k$, and $\nabla E\left(\mathbf{w}_k + \mu\mathbf{v}_k\right) \neq \mathbf{0}$.

Second, the positive definiteness of $\hat{\mathbf{B}}_{k+1}$, that is, let $\mathbf{r} \neq \mathbf{0}$ be an arbitrary vector, $\mathbf{r}^{\mathrm{T}}\hat{\mathbf{B}}_{k+1}\mathbf{r} > 0$ will be shown. Because $\hat{\mathbf{B}}_k$ is the positive definite matrix, it can be divided as $\hat{\mathbf{B}}_k = \mathbf{C}\mathbf{C}^{\mathrm{T}}$ using an arbitrary non-singular matrix $\mathbf{C}$. Let $\mathbf{t} = \mathbf{C}^{\mathrm{T}}\mathbf{r}\,(\neq \mathbf{0})$ and $\mathbf{u} = \mathbf{C}^{\mathrm{T}}\mathbf{p}_k\,(\neq \mathbf{0})$, it is shown that

$$\mathbf{r}^{\mathrm{T}}\hat{\mathbf{B}}_{k+1}\mathbf{r} = \frac{\left(\mathbf{t}^{\mathrm{T}}\mathbf{t}\right)\left(\mathbf{u}^{\mathrm{T}}\mathbf{u}\right) - \left(\mathbf{t}^{\mathrm{T}}\mathbf{u}\right)^2}{\mathbf{u}^{\mathrm{T}}\mathbf{u}} + \frac{\left(\mathbf{r}^{\mathrm{T}}\mathbf{q}_k\right)^2}{\mathbf{q}_k^{\mathrm{T}}\mathbf{p}_k} \geq 0, \tag{34}$$

with the Cauchy-Schwarz inequality [18] and the condition of (33). In (34) the equal condition is satisfied, if and only if $\left(\mathbf{t}^{\mathrm{T}}\mathbf{t}\right)\left(\mathbf{u}^{\mathrm{T}}\mathbf{u}\right) - \left(\mathbf{t}^{\mathrm{T}}\mathbf{u}\right)^2 = 0$ and $\mathbf{r}^{\mathrm{T}}\mathbf{q}_k = 0$. The former equation holds when $\mathbf{t} = \psi\mathbf{u}$ with the arbitrary scalar $\psi\,(\neq 0)$. When $\mathbf{t} = \psi\mathbf{u}$, then $\mathbf{r} = \psi\mathbf{p}_k$. Therefore, the later equation is transformed as $\mathbf{r}^{\mathrm{T}}\mathbf{q}_k = \psi\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k = 0$. This contradicts (33). Then the equal condition of (34) is not satisfied. As a result, $\hat{\mathbf{B}}_{k+1}$ holds $\mathbf{r}^{\mathrm{T}}\hat{\mathbf{B}}_{k+1}\mathbf{r} > 0$, namely positive definiteness.

□

Applying the Sherman-Morrison-Woodbury formula [18] to (30), the update formula of the inverse Hessian approximation $\hat{\mathbf{H}}_{k+1}\left(= \hat{\mathbf{B}}_{k+1}^{-1}\right)$ is given by

$$\begin{aligned}\hat{\mathbf{H}}_{k+1} = \hat{\mathbf{H}}_k &- \frac{(\hat{\mathbf{H}}_k\mathbf{q}_k)\mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k(\hat{\mathbf{H}}_k\mathbf{q}_k)^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k} \\ &+ \left(1 + \frac{\mathbf{q}_k^{\mathrm{T}}\hat{\mathbf{H}}_k\mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}\right)\frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}.\end{aligned} \tag{35}$$

From the above, it is confirmed that the NAQ has a similar convergence property with QN because $\hat{\mathbf{B}}_{k+1}$ updated by (35) holds symmetry and positive definiteness and $\hat{\mathbf{H}}_{k+1}$ is the inverse matrix of $\hat{\mathbf{B}}_{k+1}$. The update vector $\mathbf{v}_{k+1}$ of NAQ can be obtained as follow.

$$\mathbf{v}_{k+1} = \mu_k\mathbf{v}_k + \alpha_k\hat{\mathbf{c}}_k, \tag{36}$$

$$\hat{\mathbf{c}}_k = -\hat{\mathbf{H}}_k\nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k). \tag{37}$$

The momentum coefficient of μ was usually selected from value close to 1 such as $\{0.8, 0.85, 0.9, 0.95\}$ and fixed during iteration [8][17].

The limited-memory scheme can be straightly applied to the update of (36) in NAQ. The detail of the limited memory scheme is derived as follows. In the first, the update formula of (35) is transformed as

$$\begin{aligned}\hat{\mathbf{H}}_{k+1} &= \hat{\mathbf{H}}_k - \frac{(\hat{\mathbf{H}}_k\mathbf{q}_k)\mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k(\hat{\mathbf{H}}_k\mathbf{q}_k)^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k} \\ &\quad + \left(1 + \frac{\mathbf{q}_k^{\mathrm{T}}\hat{\mathbf{H}}_k\mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}\right)\frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k} \\ &= \left(\mathbf{I} - \frac{\mathbf{q}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}\right)^{\mathrm{T}}\hat{\mathbf{H}}_k\left(\mathbf{I} - \frac{\mathbf{q}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}\right) + \frac{\mathbf{p}_k\mathbf{q}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k} \qquad (38) \\ &= \hat{\mathbf{G}}_k^{\mathrm{T}}\hat{\mathbf{H}}_k\hat{\mathbf{G}}_k + \frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}, \qquad (39)\end{aligned}$$

where

$$\hat{\mathbf{G}}_k = \left(\mathbf{I} - \frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}\right). \tag{40}$$

Then $\hat{\mathbf{H}}_k$ is given by

$$\hat{\mathbf{H}}_k = \hat{\mathbf{G}}_{k-1}^{\mathrm{T}}\hat{\mathbf{H}}_{k-1}\hat{\mathbf{G}}_{k-1} + \frac{\mathbf{p}_{k-1}\mathbf{p}_{k-1}^{\mathrm{T}}}{\mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{q}_{k-1}}. \tag{41}$$

Substitute (41) into (39),

$$\begin{aligned}\hat{\mathbf{H}}_{k+1} = \hat{\mathbf{G}}_k^{\mathrm{T}}\hat{\mathbf{G}}_{k-1}^{\mathrm{T}}\hat{\mathbf{H}}_{k-1}\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k\hat{\mathbf{G}}_k^{\mathrm{T}}\frac{\mathbf{p}_{k-1}\mathbf{p}_{k-1}^{\mathrm{T}}}{\mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{q}_{k-1}}\hat{\mathbf{G}} \\ + \frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}.\end{aligned} \tag{42}$$

By repeating this operation until $k = 1$, the update formula of $\hat{\mathbf{H}}_{k+1}$ is retransformed as

$$\begin{aligned}\hat{\mathbf{H}}_{k+1} &= (\hat{\mathbf{G}}_1 \dots \hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k)^{\mathrm{T}}\hat{\mathbf{H}}_1(\hat{\mathbf{G}}_1 \dots \hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k) \\ &+(\hat{\mathbf{G}}_2 \dots \hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k)^{\mathrm{T}}\frac{\mathbf{p}_1\mathbf{p}_1^{\mathrm{T}}}{\mathbf{p}_1^{\mathrm{T}}\mathbf{q}_1}(\hat{\mathbf{G}}_2 \dots \hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k) + \dots \\ &+(\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k)^{\mathrm{T}}\frac{\mathbf{p}_{k-2}\mathbf{p}_{k-2}^{\mathrm{T}}}{\mathbf{p}_{k-2}^{\mathrm{T}}\mathbf{q}_{k-2}}(\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k)+ \\ &\hat{\mathbf{G}}_k^{\mathrm{T}}\frac{\mathbf{p}_{k-1}\mathbf{p}_{k-1}^{\mathrm{T}}}{\mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{q}_{k-1}}\hat{\mathbf{G}}_k + \frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k},\end{aligned} \tag{43}$$

where $\hat{\mathbf{H}}_1$ is an initial positive definite symmetric matrix. Since the inverse Hessian approximation $\hat{\mathbf{H}}_k$ will generally be dense, so that the cost of storing and manipulating it is prohibitive when the number of variables is large [18]. To circumvent this problem, we apply the limited-memory scheme of LQN with the user defined parameter of t to (43). The limited-memory formula of (43) between $k-th$ and $(k-t)-th$ iteration is derived as

$$\begin{aligned}\hat{\mathbf{H}}_{k+1} &= (\hat{\mathbf{G}}_{k-t+1} \dots \hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k)^{\mathrm{T}}\hat{\mathbf{H}}_k^0(\hat{\mathbf{G}}_{k-t+1} \dots \hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k) \\ &+(\hat{\mathbf{G}}_{k-t+2} \dots \hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k)^{\mathrm{T}}\frac{\mathbf{p}_{k-t+1}\mathbf{p}_{k-t+1}^{\mathrm{T}}}{\mathbf{p}_{k-t+1}^{\mathrm{T}}\mathbf{q}_{k-t+1}}(\hat{\mathbf{G}}_2 \dots \hat{\mathbf{G}}_{k-t+2}\hat{\mathbf{G}}_k) \\ &+ \dots + (\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k)^{\mathrm{T}}\frac{\mathbf{p}_{k-2}\mathbf{p}_{k-2}^{\mathrm{T}}}{\mathbf{p}_{k-2}^{\mathrm{T}}\mathbf{q}_{k-2}}(\hat{\mathbf{G}}_{k-1}\hat{\mathbf{G}}_k)+ \\ &\hat{\mathbf{G}}_k^{\mathrm{T}}\frac{\mathbf{p}_{k-1}\mathbf{p}_{k-1}^{\mathrm{T}}}{\mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{q}_{k-1}}\hat{\mathbf{G}}_k + \frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}.\end{aligned} \tag{44}$$

By substituting (44) into (37), the search vector $\hat{\mathbf{c}}_k$ of proposed LNAQ is calculated [1]. Here, $\hat{\mathbf{G}}_k$ is defined by the identity matrix and the inner products. Therefore, the search vector of LNAQ can be obtained by performing a sequence of inner products and vector summations of pairs $\{\mathbf{p}_i, \mathbf{q}_i |\ i : k-t+1, \dots, k-1, k\}$. After the new iterate $\mathbf{w}_{k+1}$ is computed, the oldest vector pair in the set of pairs $\{\mathbf{p}_i, \mathbf{q}_i\}$ is deleted and replaced by the new pairs $\{\mathbf{p}_k, \mathbf{q}_k\}$. As a result, we can derive a recursive procedure to compute $\hat{\mathbf{c}}_k$. The LNAQ scheme is illustrated in Algorithms 3 and 4. Here, Armijo's condition of (45) for LNAQ is used for the line search.

$$E(\mathbf{w}_k+\mu\mathbf{v}_k+\alpha_k\hat{\mathbf{c}}_k) \leq E(\mathbf{w}_k+\mu\mathbf{v}_k)+\hat{\chi}\alpha_k\nabla E(\mathbf{w}_k+\mu\mathbf{v}_k)^{\mathrm{T}}\hat{\mathbf{c}}_k, \tag{45}$$

where $0 < \hat{\chi} < 1$ and $\hat{\chi} = 0.001$ in this paper. Furthermore, in order to guarantees the numerical stability and the global convergence of LNAQ, (46) and (47) are added to $\mathbf{q}_k$ similarly to LQN [47].

$$\hat{\xi}_k = \omega \parallel \nabla E(\mathbf{w}_k + \mu\mathbf{v}_k) \parallel + max\{-\epsilon_k^{\mathrm{T}}\mathbf{p}_k/\parallel \mathbf{p}_k \parallel^2, 0\}, \tag{46}$$

and

$$\begin{cases}\omega = 2 & if \parallel \nabla E(\mathbf{w}_k + \mu\mathbf{v}_k) \parallel^2 > 10^{-2}, \\ \omega = 100 & if \parallel \nabla E(\mathbf{w}_k + \mu\mathbf{v}_k) \parallel^2 < 10^{-2}.\end{cases} \tag{47}$$

As a result, $\mathbf{q}_k$ is rewritten as

$$\mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k+\mu\mathbf{v}_k)+\hat{\xi}_k\mathbf{p}_k = \epsilon_k+\hat{\xi}_k\mathbf{p}_k. \tag{48}$$

Algorithm 3: The proposed LNAQ

1. $k = 1$;
2. $\mathbf{w}_1 = rand[-0.5, 0.5]$(uniform random numbers);
3. **While**$(k < k_{max})$
   (a) Calculate $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$;
   (b) Calculate the direction vector $\hat{\mathbf{c}}_k$ using Algorithm 4;
   (c) Calculate stepsize α_k using Armijo's condition;
   (d) Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mu\mathbf{v}_k + \alpha_k\hat{\mathbf{c}}_k$;
   (e) Calculate $\nabla E(\mathbf{w}_{k+1})$;
   (f) $k = k + 1$;
4. **return** $\mathbf{w}_k$;

Algorithm 4: Direction Vector of LNAQ

1. $\hat{\mathbf{c}}_k = -\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$;
2. for $i : k, k-1, \dots, k-\min(k, (t-1))$;
   (a) $\hat{\beta}_i = \mathbf{p}_i^{\mathrm{T}}\hat{\mathbf{c}}_k/\mathbf{p}_i^{\mathrm{T}}\mathbf{q}_i$;
   (b) $\hat{\mathbf{c}}_k = \hat{\mathbf{c}}_k - \hat{\beta}_i\mathbf{q}_i$;
3. if $k > 1, \hat{\mathbf{c}}_k = (\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k/\mathbf{q}_k^{\mathrm{T}}\mathbf{q}_k)\hat{\mathbf{c}}_k$;
4. for $i : k-\min(k, (t-1)), \dots, k-1, k$;
   (a) $\hat{\tau} = \mathbf{q}_i^{\mathrm{T}}\hat{\mathbf{c}}_k/\mathbf{q}_i^{\mathrm{T}}\mathbf{p}_i$;
   (b) $\hat{\mathbf{c}}_k = \hat{\mathbf{c}}_k - (\hat{\beta}_i - \hat{\tau})\mathbf{p}_i$;
5. **return** $\hat{\mathbf{c}}_k$;

In Algorithm 3, two times calculations of the gradient vectors of $\nabla E(\mathbf{w}_k+\mu\mathbf{v}_k)$ and $\nabla E(\mathbf{w}_{k+1})$ were needed within a training loop whereas LQN needs one derivation of the gradient. This is a disadvantage of LNAQ, but the algorithm can further shorten the iteration counts to cancel out the effect of this shortcoming [1]. The simulation results will show the above fact.

V. Simulation results

Computer simulations are conducted in order to demonstrate the validity of the proposed LNAQ. In the simulations the feedforward neural networks with a hidden layer and an arbitrary number of hidden layer's neurons were used. Each neuron has a sigmoid function as $sig(x) = 1/(1 + exp(-x))$. The performance of LNAQ is compared with conventional algorithms such as BP [2], CM [8], NAG [8], AdaGrad [9], RMSprop [12], Adam [14] and LQN [18] for two benchmark problems. Benchmark problems used here are a function approximation problem of Levy function [48] and a microwave circuit modeling problem of low-pass filter [17][36]-[38][49]. Ten independent runs were performed with different starting values of $\mathbf{w}$, which are initialized by uniform random numbers within $[-0.5, 0.5]$. Each hyper-parameter of AdaGrad, RMSprop and Adam is set to the default value of each original

paper, respectively. These adaptive methods are mainly utilized in the stochastic (mini-batch) mode. However, the problems in this paper need the full batch method [15]. Therefore, the full batch scheme is applied to all algorithms. The momentum coefficient of μ used in CM, NAG and LNAQ are 0.8, 0.85, 0.9 and 0.95 as [8][17]. The simulations were performed on the computer, which has Intel Core i7-8700 3.2GHz processor and 8GB memory. Each trained neural network was estimated by the average, best and worst of $E(\mathbf{w})$, the average of computational time (s) and the average of iteration counts (k). Each element of the input and desired vectors of T_r is normalized within $[-1.0, 1.0]$ in the simulations.

A. *Levy function approximation problem*

Levy function ($\mathrm{R}^n \rightarrow \mathrm{R}^1$) shown in (49) is used for the first function approximation problem. The Lavy function is a multimodal function with highly-nonlinear characteristic. Therefore, the function usually used as a benchmark problem for the multimodal function optimization [48].

$$f(x_1 \dots x_n) = \frac{\pi}{n}\left\{ \sum_{i=1}^{n-1}[(x_i-1)^2(1+10\sin^2(\pi x_{i+1}))] + 10\sin^2(\pi x_1) + (x_n-1)^2 \right\}, x_i \in [-4,4], \forall i, \tag{49}$$

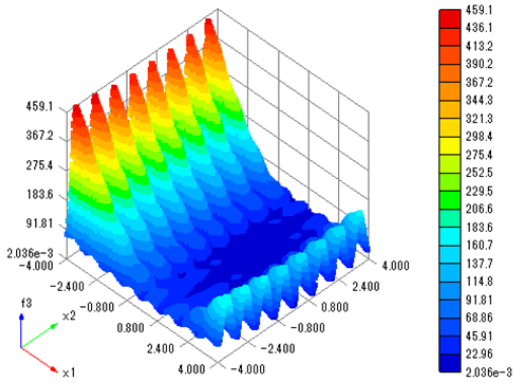

Figure 1. Levy Function $f(x_1, x_2)$.

where n denotes the input dimension. In Figure 1, Levy function with $n = 2$ dimensions of (49) is shown. From the figure, we can obtain the highly-nonlinear characteristic of the function. In this simulation the dimension of input vector $\mathbf{x}$ is set to $n = 5$. The inputs and an output are $x_1, \cdots, x_5$ and $f(x_1, \cdots, x_5)$, respectively. The trained network has a hidden layer with 50 hidden neurons. Therefore, the structure of neural network is 5-50-1 and the dimension of $\mathbf{w}$ is 351. The number of training data is $|T_r| = 5000$, which are generated by uniformly random number in $x_i \in [-4, 4]$. Maximum number of iteration is set to $k_{max} = 2 \times 10^4$. Here, we verified LNAQ from the viewpoints of two kinds of comparisons. First one is the comparison with LQN for iteration and computer time. Second, the proposed LNAQ is compared with the conventional algorithms for the training errors.

1) *Comparison of LQN and LNAQ*

Here, we compare LNAQ and LQN with respect to iteration count (k) and the computational time (s) for several memories. The range of storage memory t is from 10 to 100 at intervals of 10. The terminate conditions are set to $E(\mathbf{w}) \leq 1.0 \times 10^{-4}$. For function approximation problems, the small MSE of $E(\mathbf{w})$ is very important, because the trained network with the small $E(\mathbf{w})$ can become an accurate neural network model. Therefore, the average iteration counts and computational times until $E(\mathbf{w}) \leq 1.0 \times 10^{-4}$ within $k_{max} = 2 \times 10^4$ are obtained

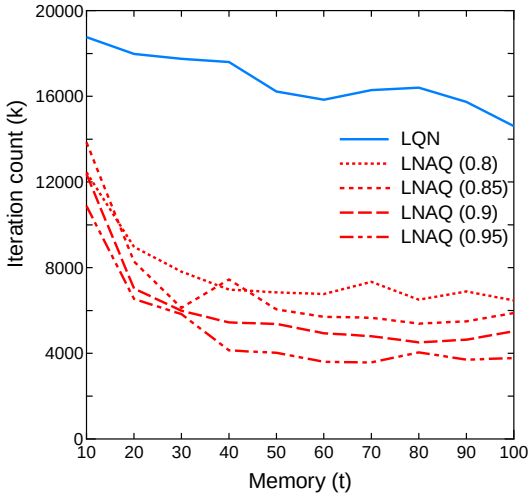

Figure 2. The average of iteration count vs memories.

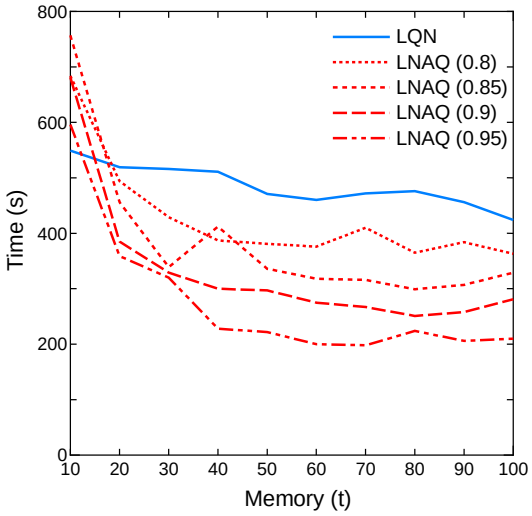

Figure 3. The average of calculation time vs memories.

in this comparison. Figure 2 shows the average of iteration count (k) vs memories (t) for LQN and LNAQ. From this figure, it can be seen that LNAQ converges with less iteration count than LQN regardless of μ. Therefore, LNAQ has the

TABLE I. Summary of Levy function.

Algorithm	μ	Memory	$E(\mathbf{w})(\times 10^{-3})$ Ave/Best/Worst	Time (s)	Time / Iteration (ms)
BP	-	-	38.8 / 30.6 / 52.0	487	24.35
CM	0.9	-	32.4 / 0.31 / 54.8	806	40.30
	0.95	-	47.5 / 13.4 / 54.8	854	42.70
NAG	0.9	-	55.0 / 54.8 / 55.7	802	40.10
	0.95	-	55.0 / 54.8 / 56.2	738	36.90
AdaGrad	-	-	52.5 / 52.3 / 53.1	488	24.40
RMSprop	-	-	43.0 / 33.2 / 54.1	487	24.35
Adam	-	-	1.20 / 0.16 / 10.2	487	24.35
LQN	-	30	0.0710 / 0.0338 / 0.167	732	36.60
	-	40	0.0679 / 0.0302 / 0.143	732	36.60
LNAQ	0.9	30	0.0174 / 0.00517 / 0.0448	1,210	60.50
	0.95	30	0.0295 / 0.00578 / 0.145	1,230	61.50
LNAQ	0.9	40	0.0205 / 0.00529 / 0.0407	1,220	61.00
	0.95	40	0.0256 / 0.00585 / 0.0583	1,250	62.50

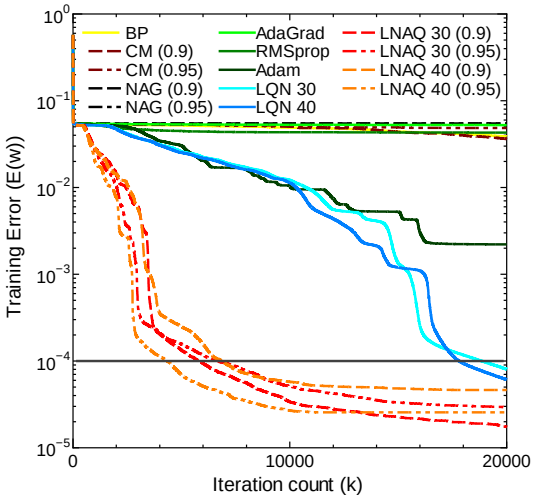

Figure 4. The average training errors vs iteration count of Levy function.

ability to significantly reduce the iteration counts compared to LQN. Furthermore, it is shown that the decrease of iteration is hardly seen in memory (t) larger than 40 and the momentum coefficients μ closer to 1, that is $\mu = 0.9$ and 0.95 converge faster. However, LNAQ requires two calculations of gradient in one iteration. That is, it takes time to compare with LQN in one iteration. Therefore, it is necessary to compare the calculation time until the training end. Figure 3 shows the average of calculation times (s) vs memories (t) for LQN and LNAQ. From Figure 3, it can be seen that LNAQ is inferior to LQN in terms of calculation time, when the storage memory of (t) is small. However, when t increases, it is easy to conclude that LNAQ is faster than LQN. From these figures, it is confirmed that memories of $t = 30$ or 40 and coefficients of $\mu = 0.9$ and 0.95 are recommended.

2) Comparison of LNAQ and conventional algorithms

In these simulations, the proposed LNAQ is compared with BP, CM, NAG, AdaGrad, RMSprop, Adam and LQN. The storage amount of memories is experimentally set to $t = 30$ and 40 from the above results. The momentum coefficients μ of CM, NAG and LNAQ are set to $\mu = 0.9$ and 0.95. Here, the terminate condition is set to $k_{max} = 2 \times 10^4$. This means that the iteration continues after $E(\mathbf{w}) \leq 1.0 \times 10^{-4}$. The summary of results is shown in Table 1 and the average of training errors of BP, CM, NAG, AdaGrad, RMSprop, Adam, LQN and LNAQ for the iteration count is illustrated in Figure 4. From Figure 4 and Table 1, The conventional algorithms based on the first order methods such as BP, CM, NAG, AdaGrad and RMSprop could not converge to small training errors. From Table 1, it is confirmed that Adam, LQN and LNAQ converge to small errors depending on the initial value. In comparing of the average training errors, LQN and LNAQ can obtain the small average errors compared with Adam. Especially, the average error of LNAQ ($t = 30, \mu = 0.9$) is smallest and almost the same as the worst error. These results show the robustness with respect to the initial value. On the other hand, the calculation time of LQN is faster than the one of LNAQ for the same iteration count ($k_{max} = 2 \times 10^4$). This is caused by the drawback namely two times calculations of the gradients of LNAQ. However, LNAQ can reach the small training error ($E(\mathbf{w}) \leq 1.0 \times 10^{-4}$) faster than LQN. This fact can be confirmed in Figure 4.

B. *Microwave circuit modeling of low-pass filter*

Neural networks can be trained using measured or simulated microwave device data such as EM and physical data [3]-[6]. The trained neural networks can be used as models of microwave devices in place of CPU-intensive EM/physics models to significantly speed up circuit design while maintaining EM/physics-level accuracies [3][5]. Neural network based modeling has been used to model a variety of microwave circuit components at both device and circuit levels. In this simulation, we applied LNAQ to develop a neural network model of the microstrip low-pass filter (LPF) [17][36]-[38] illustrated in Figure 5. The dielectric constant and height of

TABLE II. Summary of LPF.

Algorithm	μ	Memory	$E(\mathbf{w})(\times 10^{-3})$ Ave/Best/Worst	Time (s)	Time/ Iteration (ms)
BP	-	-	22.4 / 19.6 / 24.3	143	2.86
CM	0.9	-	23.7 / 9.56 / 29.2	264	5.28
	0.95	-	24.0 / 6.41 / 29.2	277	5.54
NAG	0.9	-	106 / 16.3 / 832	248	4.96
	0.95	-	26.2 / 15.5 / 29.2	247	4.94
AdaGrad	-	-	25.3 / 24.8/ 25.6	142	2.84
RMSprop	-	-	26.6 / 26.2 / 27.0	144	2.88
Adam	-	-	5.54 / 4.66 / 6.35	142	2.84
LQN	-	30	6.89 / 5.81 / 7.68	246	4.92
	-	40	7.08 / 6.42 / 7.81	247	4.94
LNAQ	0.9	30	2.15 / 1.59 / 3.13	378	7.56
	0.95	30	1.63 / 1.36 / 2.06	377	7.54
LNAQ	0.9	40	1.97 / 1.57 / 2.80	380	7.60
	0.95	40	1.49 / 1.23 / 1.93	377	7.54

the substrate of LPF are 9.3mm and 1mm, respectively. The length D ranges 12-20mm at intervals of 1mm. The frequency range was 0.1 to 4.5GHz. Each set of contains 221 samples. The inputs of the neural network, x_1 and x_2 are frequency f and length D in which training data T_r and test data T_e are set to $D = [12, 14, 16, 18, 20]$ mm and $[13, 15, 17, 19]$ mm, respectively. The outputs, o_1 and o_2 are the magnitudes of S-parameters, $|S_{11}|$ and $|S_{21}|$, respectively. These data are obtaining by the standard software of *sonnet* [49]. Training data is illustrated in Figure 6. As shown in Figure 6, there are many irregularly aligned poles in S-parameters and the modeling of the poles is the most important in microwave circuit problems. Therefore the microwave circuit modeling is a strong nonlinear problem and needs very small training and testing errors. The number of hidden neurons is 45. Therefore, the structure of neural network is 2-45-2 and the dimension of $\mathbf{w}$ is 227. The maximum iteration count is set to $k_{max} = 5 \times 10^4$. The purposed LNAQ is also compared with BP, CM, NAG, AdaGrad, RMSprop, ADAM and LQN. Memories (t) are selected 30 and 40 for both of LQN and LNAQ. The coefficients of μ for CM, NAG and LNAQ are set to 0.9 and 0.95. The summary of results is shown in Table 2 and the training errors for iteration counts are illustrated in Figure 7. From Table 2 and Figure 7, the first-order methods such as BP, CM, NAG, AdaGrad and RMSprop could not obtain the small training errors for the practical methods. Adam can obtain the small training errors compared with LQN for this problem. In comparison of Adam with LNAQ, LNAQ need more computational time than Adam because of two calculations of gradient and the complex procedure for the calculation of the direction. However, the proposed LNAQ can converge to smaller value of training error than Adam and LQN. Especially, the average training error of LNAQ ($t = 40$ and $\mu = 0.95$) can converge to 1.49×10^{-3}. Furthermore, LNAQ can obtain the small difference between the best and the worst errors. This means that LNAQ is also robust with respect to the initial value for this problem.

For measuring accuracy of modeling, the outputs of the trained neural model for $D = [13, 15, 17, 19]$ mm, which are not included in training are compared with the test data of $|S_{11}|$ and $|S_{21}|$. The trained models are selected from neural networks trained by Adam and LNAQ ($t = 40$ and $\mu = 0.95$) with the smallest training errors 4.66×10^{-3} and 1.23×10^{-3}, respectively. The test errors $E_{test}(\mathbf{w})$ obtained by Adam and LNAQ are 3.15×10^{-3} and 0.656×10^{-3}, respectively. Figure 8 and 9 shows the comparison of the test data of $D = 13$

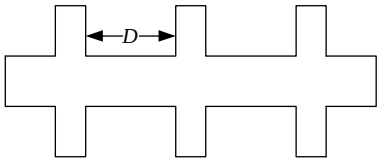

Figure 5. Layout of LPF.

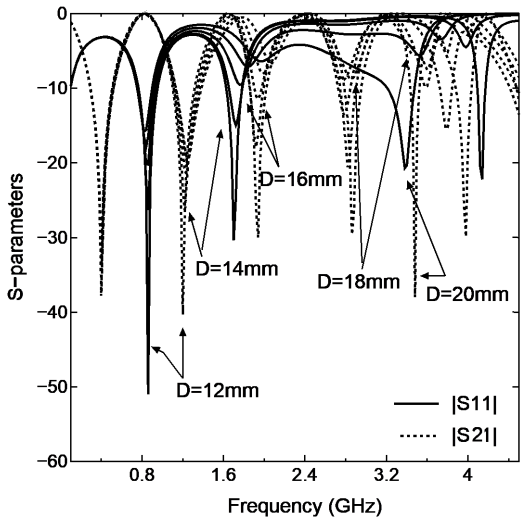

Figure 6. Training data set for LPF.

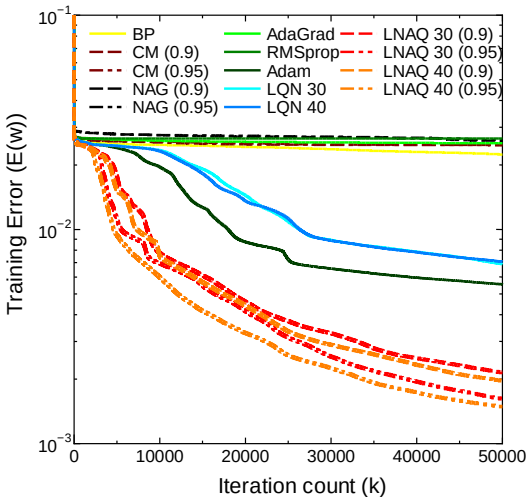

Figure 7. The average training errors vs iteration count of LPF.

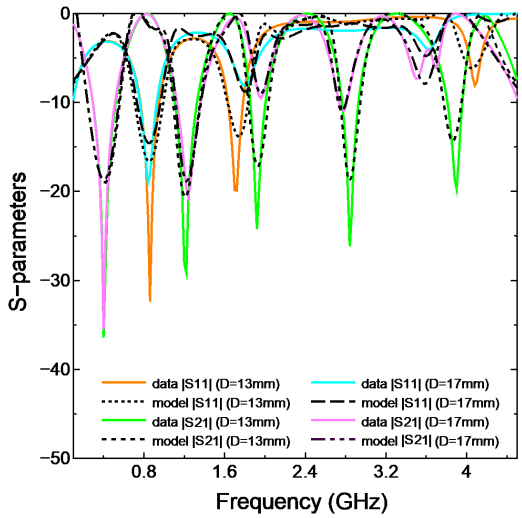

Figure 8. Example of comparison between test data and neural model trained by Adam.

mm and 17 mm with the model outputs trained by Adam and LNAQ, respectively. It can be seen from Figure 8 of the model trained by Adam that there are multiple large gaps between the model outputs and test data. They are prominent in places of pole. On the other hand, it can be confirmed that the neural model trained by LNAQ and the test data are showing good match between them from Figure 9.

VI. Conclusion

In this paper, we proposed a novel training algorithm called LNAQ, which was developed based on Limited-memory method of QN incorporating the momentum acceleration scheme and Nesterov's accelerated gradients vector. The effectiveness of the proposed LNAQ was demonstrated through the computer simulations compared with the conventional algorithms such as BP, CM, NAG, AdaGrad, RMSprop, Adam and LQN. For highly-nonlinear problem, the first-order methods such as BP, CM, NAG, AdaGrad and RMSprop could not obtain desired small training errors for the function approximation and the microwave circuit modeling problems. Only Adam could get small errors depending on the problem and the initial value of $\mathbf{w}$. On the other hand, the curvature information-based method such as LQN and LNAQ could obtain the small errors for a function approximation problem. For a real-world problem of microwave circuit modeling the efficient and practical models could be trained by only LNAQ. Furthermore, the effectiveness of the momentum coefficient for QN with the limited memory scheme was demonstrated through the results of the training errors for iteration. This means that LNAQ can reduce training errors earlier than other method. LNAQ may take time to obtain a solution compared to other methods because of its drawback. Depending on the problem, however, it may be the only algorithm that can get a practical model that cannot be obtained by the other methods. This is very important issue for modeling of highly-nonlinear problems.

In the future the momentum parameters μ will be studied. This parameter was analytically determined for the first-order method of NAG in [7] for the convex problems whereas the fixed values were used in [8][16] for the neural training problems of the non-convex problems in the same way as

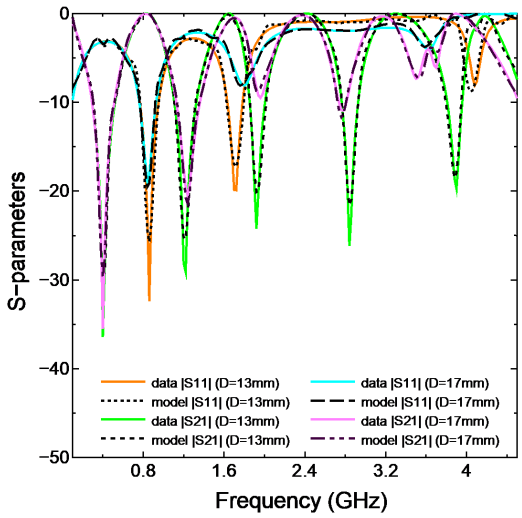

Figure 9. Example of comparison between test data and neural model trained by LNAQ.

this paper. Therefore, the analytical studies of the momentum parameter for the second-order method of LNAQ will be done in the future. Furthermore, the validity of the proposed algorithm for more highly-nonlinear function approximation problems and the much huge scale problems including deep networks will be demonstrated.

Acknowledgment

The authors thank to Prof. Q.J. Zhang at Carleton University, Canada, for his support of microwave circuit models. This work was supported by Japan Society for the Promotion of Science (JSPS), KAKENHI (17K00350).

References

[1] S. Mahboubi and H. Ninomiya, "A novel training algorithm based on limited-memory quasi-Newton method with Nesterov's accelerated gradient for neural networks," *IARIA / FUTURE COMPUTING'18*, pp. 1−3, Feb. 2018.

[2] S. Haykin, "*Neural Networks and Learning Machines 3^{rd}*," Pearson, 2009.

[3] Q. J. Zhang, K. C. Gupta, and V. K. Devabhaktuni, "Artifical neural networks for RF and microwave design-from theory to practice," *IEEE Trans. Microwave Theory and Tech.*, vol.51, pp.1339−1350, Apr. 2003.

[4] H. Ninomiya , "A hybrid global/local optimization technique for robust training and its application to microwave neural network models," *J.Signal Procassing*, 14, 3, pp.213−222, 2010.

[5] H. Kabir, L. Zhang, M. Yu, P. H. Aaen, J. Wood, and Q. J. Zhang, "Smart modeling of microwave devices," *IEEE Microwave Magazine*, vol.11, no.3, pp.105−118, May. 2010.

[6] H. Ninomiya, S. Wan, H. Kabir, X. Zhang and Q. J. Zhang, "Robust training of microwave neural network models using combined global/local optimization techniques," *IEEE MTT-S International Microwave Symposium (IMS) Digest*, pp.995−998, Jun, 2008.

[7] Y. Nesterov, "*Introductory Lectures on Convex Optimization: A Basic Course*," 2004.

[8] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," *ICML'13*, 2013.

[9] D. John, H. Elad, and S.Yoram, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, pp.2121−2159, Jul. 2011.

[10] S. Ruder, "An overview of gradient descent optimization algorithm," arXiv preprint arXiv:1609.04747. 2016.

[11] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," arXiv preprint arXiv:1212.5701. 2012.

[12] T. Tieleman and G. Hinton, "Lecture 6.5 - RMSProp," *COURSERA: Neural Networks for Machine Learning. Technical report*, 2012.

[13] M. Riedmiller and H. Braun, "Rprop - A fast adaptive learning algorithm," *ISCIS VII*, 1992.

[14] P. D. Kinguma and J. Ba, "Adam: A method for stochastic optimization," *ICLR'15*, vol.5, May. 2015.

[15] H. Ninomiya, "Dynamic sample size selection based quasi-Newton training for highly nonlinear function approximation using multilayer neural networks," *IEEE&INNS IJCNN'13*, pp.1932−1937, Aug. 2013.

[16] A. Basu, S. De, A. Mukherjee, and E. Ullah, "Convergence guarantees for RMSProp and ADAM in non-convex optimization and their comparison to Nesterov acceleration on autoencoders.", arXiv:1807.06766v1, Jul. 2018.

[17] H. Ninomiya, "A novel quasi-Newton optimization for neural network training incorporating Nesterov's accelerated gradient," *IEICE NOLTA Journal*, vol.E8-N, no.4, pp.289−301, Oct. 2017.

[18] J. Nocedal and S. J. Wright, "*Numerical Optimization Second Edition*," Springer, 2006.

[19] W. Forst and D. Hoffmann, "*Optimization - Theory and Practice*," Springer, 2010.

[20] I. Goodfellow, Y. Bengio, and A. Courville. "*Deep learning (adaptive computation and machine learning series)*," Adaptive Computation and Machine Learning series, 2016.

[21] J. Wood and D. E. Root, eds,"*Fundamentals of Nonlinear Behavioral Modeling for RF and Microwave Design*," ARTECH HOUSE, 2005.

[22] Q. J. Zhang, J, Bandler, S. Koziel H. Kabir, and L, Zhang, "ANN and space mapping for microwave modeling and optimization," *IEEE MTT-S International*, pp.980−983, May. 2010.

[23] R. M. Hassani, D, Haerle, and R. Grosu, "Efficient modeling of complex analog integrated circuits using neural networks," *IEEE PRIME'12*, pp.1−4, Jun. 2016.

[24] J. Michel and Y. Herve, "VHDL-AMS behavioral model of an analog neural networks based on a fully parallel weight perturbation algorithm using incremental on-chip learning," *IEEE International Symposium on Industrial Electronics*, vol.1, pp. 211−216, May. 2004.

[25] A. Jafari, S. Sadri, and M. Zekri, "Design optimization of analog integrated circuits by using artificial neural networks," *IEEE SoCPaR*, Nov. 2010.

[26] R. M. Hasani, D. Haerle, C. F. Baumgartner, A. R. Lomuscio, and R. Grosu, "Compositional neural-network modeling of complex analog circuits," *IEEE IJCNN'17*, pp.2235−2242, May. 2017.

[27] M. Kraemer, D. Dragomirescu, and Robert Plana, "A novel technique to create behavioral models of differential oscillators in VHDL-AMS," *SM2ACD*, Oct. 2008.

[28] M. Kraemer, D. Dragomirescu, and R. Plana, "Nonlinear behavioral modeling of oscillators in VHDL-AMS using artificial neural networks," *IEEE RFIC*, pp.689−692, Jun. 2008.

[29] J. P. Garcia, F. Q. Pereira, D. C. Rebenaque, J. L. G. Tornero, and A. A. Melcon, "A neural-network method for the analysis of multilayered shielded microwave circuits," *Proc.IEEE Trans. Microwave Theory Tech.*, vol.54, no.1, pp.309−320, Jan. 2006.

[30] V. Rizzoli, A. Costanzo, D. Masotti, A. Lipparini, and F. Mastri, "Computer-aided optimization of nonlinear microwave circuits with the aid of electromagnetic simulation," *IEEE Trans. Microwave Theory Tech.*, vol.52, no.1, pp.362−377, Jan. 2004.

[31] V. Rizzoli, A. Neri, D. Masotti, and A. Lipparini, "A new family of neural network-based bidirectional and dispersive behavioral models for nonlinear RF/microwave subsystems," *Int. J. RF Microwave Comput.-Aided Eng.*, vol.12, no.1, pp.51−70, Jan. 2002.

[32] G. L. Creech, B. J. Paul, C. D. Lesniak, T. J. Jenkins, and M. C. Calcatera, "Artificial neural networks for fast and accurate EM-CAD of microwave circuits," *IEEE Trans. Microwave Theory Tech.*, vol.45, no.5, pp.794−802, May. 1997.

[33] Y. Wang, M. Yu, H. Kabir, and Q. J. Zhang, "Effective design of cross-coupled filter using neural networks ad coupling matrix," *IEEE MTT-S Int. Microwave Symp. Dig.*, pp.1431−1434, Jun. 2006.

[34] H. Kabir, Y. Wang, M. Yu, and Q. J. Zhang, "Neural network inverse modeling and applications to microwave filter design," *IEEE Trans. Microwave Theory Tech.*, vol.56, no.4, pp.867−879, Apr. 2008.

[35] M. M. Vai, S. Wu, B. Li, and S. Prasad, "Reverse modeling of microwave circuits with bidirectional neural network models," *IEEE Trans. Microwave Theory Tech.*, vol.46, pp.1492−1494, Oct. 1998.

[36] H. Ninomiya, "Microwave neural network models using improved online quasi-Newton training algorithm," *Journal of Signal Processing*, vol.15, no.6, pp.483−488, Nov. 2011.

[37] H. Sharma and Q. J. Zhang, "Transient electromagnetic modeling using recurrent neural network," *IEEE MTT-SIMS Digest*, pp.1597−1600, Jun. 2005.

[38] W. J. R. Hoefer and P. P. M. So, "*The MEFiSTo-2D Theory*," Victoria, BC, Canada: Faustus Scientific Corporation, 2001.

[39] T. Liu, S. Boumaiza, and F. M. Ghannouchi, "Dynamic behavioral modeling of 3G power amplifier using real-valued time delay neural networks," *IEEE Trans. Microwave Theory Tech.*, vol.52, no.3, pp.1025−1033, Mar. 2004.

[40] M. Isaksson, D. Wisell, and D. Ronnow, "Wide-band modeling of power amplifiers using radial-basis function neural networks," *IEEE Trans. Microwave Theory Tech.*, vol. 53, no. 11, pp. 3422−3428, Nov. 2005.

[41] B. O'Brien, J. Dooley, and T. J. Brazil, "RF power amplifier behavioral modeling using a globally recurrent neural network," *IEEE MTT-S Int. Microwave Symp. Dig*, pp.1089−1092, Jun. 2006.

[42] J. Wood, D. E. Root, and N. B. Tufillaro, "A behavioral modeling approach to nonlinear model-order reduction for RF/microwave ICs and systems," *IEEE Trans. Microwave Theory Tech.*, vol.52, no.9, pp.2274−2284, Sep. 2004.

[43] P. M. Watson and K. C. Gupta, "EM-ANN models for microstrip vias and interconnects in dataset circuits," *IEEE Trans. Microwave Theory Tech.*, vol.44, no.12, pp.2495−2503, Dec. 1996.

[44] S. Roweis, "Levenberg-marquardt optimization," *Notes, University Of Toronto*, 1996.

[45] M. K. Transtrum and J. P. Sethna. "Improvements to the Levenberg-Marquardt algorithm for nonlinear least-squares minimization," arXiv preprint arXiv:1201.5885, Jan. 2012.

[46] M. I. A. Lourakis, "A brief description of the Levenberg-Marquardt algorithm implemented by levmar," *Foundation of Research and Technology*, pp.1−6, Feb. 2005.

[47] D. H. Li and M. Fukushima, "A modified BFGS method and its global convergence in nonconvex minimization," *Journal of Computational and Applied Mathematics*, vol.129, pp.15−35, 2001.

[48] D. Gao, N. Ruan, and W. Xing, editors, "*Advances in Global Optimization*," Springer Proceedings in Mathematics & Statistics, 2014.

[49] *Sonnet*, Full-wave 3D Planar Electromagnetic Field Solver Software for High Frequency EM Simulation, Sonnet Software, Inc.